UDACITY

< Return to Classroom

# Explore US Bikeshare Data

REVIEW

CODE REVIEW 7

HISTORY

**bikeshare.py** 7

```
1  import time
2  import pandas as pd
3  import numpy as np
4  import calendar
5
6  CITY_DATA = { 'chicago': 'chicago.csv',
7                'new york city': 'new_york_city.csv',
8                'washington': 'washington.csv' }
9
10 def get_filters():
11     """
12     Asks user to specify a city, month, and day to analyze.
```

```
    Returns:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no month filter
        (str) day - name of the day of week to filter by, or "all" to apply no day filter
    """
    print('Let\'s explore some US bikeshare data!')
    # get user input for city (chicago, new york city, washington). HINT: Use a while loop to handle invalid inputs

    cities = ('chicago', 'new york city', 'washington')
    city = ''
    while city not in cities:
        city = input('\nWould you like to see data for Chicago, New York City, or Washington?\n').lower()
```

AWESOME

## First Class! 🚀

Good job using `.lower()` to handle case-sensitivity while capturing user inputs! 🌟

```
        if city not in cities:
            print('\nYou have entered an invalid CITY. Please select from Chicago, New York City or Washington.')
            continue
        else:
            break

    # get user input for month (all, january, february, ... , june)
    months = ('all', 'january', 'february', 'march', 'april', 'may', 'june')
    month = ''
    while month not in months:
        month = input('\nWhich month would you like to filter the data by: January, February, March, April, May, or June?
        if month not in months:
            print('\nYou have entered an invalid MONTH. Please select from January, February, March, April, May, June or a
            continue
        else:
            break

    # get user input for day of week (all, monday, tuesday, ... sunday)
    days = ('all', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday')
    day = ''
    while day not in days:
        day = input('\nWhich day would you like to filter the data by: Monday, Tuesday, Wednesday, Thursday, Friday, Satur
```

```
        if day not in days:
            print('\nYou have entered an invalid DAY. Please select from Monday, Tuesday, Wednesday, Thursday, Friday, Sat
            continue
        else:
            break

    print('-'*120)
    return city, month, day


def load_data(city, month, day):
    """
    Loads data for the specified city and filters by month and day if applicable.

    Args:
        (str) city - name of the city to analyze
        (str) month - name of the month to filter by, or "all" to apply no month filter
        (str) day - name of the day of week to filter by, or "all" to apply no day filter
    Returns:
        df - Pandas DataFrame containing city data filtered by month and day
    """
    # load data file into a dataframe
    df = pd.read_csv(CITY_DATA[city])

    # convert the Start Time column to datetime
    df['Start Time'] = pd.to_datetime(df['Start Time'])

    # extract month from the Start Time column to create a month column
    df['Month'] = df['Start Time'].dt.month

    # extract day from the Start Time column to create a day_of_week column
    df['Day_of_week'] = df['Start Time'].dt.day_name()

    # filter by month if applicable and create new dataframe
    if month != 'all':
        months = ['january', 'february', 'march', 'april', 'may', 'june']
        month = months.index(month) + 1
        df = df[df['Month'] == month]

    # filter by day of week if applicable and create new dataframe
    if day != 'all':
        df = df[df['Day_of_week'] == day.title()]

    return df


```

```
93 def time_stats(df):
94     """Displays statistics on the most frequent times of travel."""
95
96     print('\nCalculating The Most Frequent Times of Travel...\n')
97     start_time = time.time()
98
99     # display the most common month
100    common_month = df['Month'].mode()[0]
101    print('Most popular month of traveling:', calendar.month_name[common_month])
```

AWESOME

## Amazing! 🚀

Excellent work displaying the name of the most common month. Most student submissions just leave it at displaying the month number, but you hav

```
102
103    # display the most common day of week
104    common_day = df['Day_of_week'].mode()[0]
105    print('Most popular day of traveling:', common_day)
106
107    # display the most common start hour
108    df['Hour'] = df['Start Time'].dt.hour
109    common_hour = df['Hour'].mode()[0]
110    print('Most popular hour of the day to start traveling:', common_hour)
111
112    print("\nThis took %s seconds." % (time.time() - start_time))
113    print('-'*120)
114
115
116 def station_stats(df):
117    """Displays statistics on the most popular stations and trip."""
118
119    print('\nCalculating The Most Popular Stations and Trip...\n')
120    start_time = time.time()
121
122    # display most commonly used start station
123    common_start = df['Start Station'].mode()[0]
```

AWESOME

## Great Work! 🚀

Well done using `.mode()` to calculate the most common Start Station!

### Suggestion 💡

An improvement that you can make when displaying the "mode" type statistics is to display the count of occurrences of the most common item. You

```
124     print('Most commonly used start station:', common_start)
125
126     # display most commonly used end station
127     common_end = df['End Station'].mode()[0]
128     print('Most commonly used end station:', common_end)
129
130     # display most frequent combination of start station and end station trip
131     df['Frequent Trip'] = df['Start Station'] + ' to ' + df['End Station']
```

AWESOME

## Excellent! 🚀

Good work using this to calculate the combination of start and stop stations. 🌟

### Alternatively... 😉

You can also use Pandas' groupby on both Start and End Station columns, and sum up the occurrences to achieve the same effect. Just another way c

Here's the code for getting the most popular trip through group bys:

```
most_popular_trip = df.groupby(['Start Station', 'End Station']).size().idxmax()
```

```
132     common_trip = df['Frequent Trip'].mode()[0]
133     print('Most common trip from start station to end station:', common_trip)
134
135     print("\nThis took %s seconds." % (time.time() - start_time))
136     print('-'*120)
```

```
137
138
139 def trip_duration_stats(df):
140     """Displays statistics on the total and average trip duration."""
141
142     print('\nCalculating Trip Duration...\n')
143     start_time = time.time()
144
145     # display total travel time
146     total_travel_time = df['Trip Duration'].sum()
```

AWESOME

## Nicely done! 🚀

Good job using `.sum()` to calculate the total trip duration.

### Suggestion 💡

You can go one step further and display the trip duration in days/hours/minutes etc, so that it is easily understandable by users of your program.

```
147     print('Total travel time:', total_travel_time, 'seconds')
148
149     # display mean travel time
150     mean_travel_time = df['Trip Duration'].mean()
151     print('Average travel time:', mean_travel_time, 'seconds')
152
153     print("\nThis took %s seconds." % (time.time() - start_time))
154     print('-'*120)
155
156
157 def user_stats(df):
158     """Displays statistics on bikeshare users."""
159
160     print('\nCalculating User Stats...\n')
161     start_time = time.time()
162
163     # Display counts of user types
164     user_type_count = df['User Type'].value_counts()
165     print('User Type Count:\n',user_type_count)
```

```
166
167     # Display counts of gender
168     try:
```

AWESOME

## Well Done! 🚀

Good job accounting for the missing data in the Washington dataset. 🌟

You've also thought through the user experience and displayed a message to the user, notifying them If data doesn't exist in the dataframe.

```
169         gender_count = df['Gender'].value_counts()
170         print('\nGender Count:\n', gender_count)
171     except KeyError:
172         print('\nGender Count: data not available.')
173
174     # Display earliest, most recent, and most common year of birth
175     try:
176         birth_min = int(df['Birth Year'].min())
177         print('\nEarliest year of birth:', birth_min)
178     except KeyError:
179         print('\nEarliest birth year: data not available.')
180
181     try:
182         birth_max = int(df['Birth Year'].max())
183         print('Most recent year of birth:', birth_max)
184     except KeyError:
185         print('\nMost recent birth year: data not available.')
186
187     try:
188         birth_mode = int(df['Birth Year'].mode()[0])
189         print('Most common year of birth:', birth_mode)
190     except KeyError:
191         print('\nMost common birth year: data not available.')
192
193     print("\nThis took %s seconds." % (time.time() - start_time))
194     print('-'*120)
195
196
197 def display_data(df):
```

AWESOME

# Splendid! 🚀

A simple and succinct function to display the raw data to the user interactively using df.iloc, together with a while loop. Very well done!

## Suggestion 💡

You may have noticed that not all columns are displayed when printing raw data. You can use `pd.set_option('display.max_columns',200)` at the sta your dataframe when printing.

```
198     """Displays raw data 5 rows at a time, if requested."""
199
200     view_data = input('\nWould you like to view 5 rows of raw data? yes or no:\n').lower()
201     if view_data != 'no':
202         start_loc = 0
203         while (start_loc < df['Start Time'].count() and view_data != 'no'):
204             print(df.iloc[start_loc:start_loc+5])
205             start_loc += 5
206             more_data = input('\nWould you like to view 5 more rows of data? yes or no:\n').lower()
207             if more_data != 'yes':
208                 break
209
210
211 def main():
212     while True:
213         city, month, day = get_filters()
214         df = load_data(city, month, day)
215
216         time_stats(df)
217         station_stats(df)
218         trip_duration_stats(df)
219         user_stats(df)
220         display_data(df)
221
222         restart = input('\nWould you like to restart? Enter yes or no.\n')
223         if restart.lower() != 'yes':
224             break
225
```

```
226
227 if __name__ == "__main__":
228     main()
229
```

RETURN TO PATH

**Rate this review**

START